Informatik and Mathematical Modelling 26/11 07
DTU Michael R. Hansen

Problem A

Consider the following declarations:

```
fun map f []      = []
  | map f (x::xs) = (f x) :: map f xs

infix @ fun [] @ ys      = ys
  | (x::xs) @ ys = x :: (xs @ ys)
```

Prove

$$\texttt{map}\, f\, (l_1 \,@\, l_2) \;=\; (\texttt{map}\, f\, l_1) \,@\, (\texttt{map}\, f\, l_2) \tag{1}$$

The proof is by structural induction over the list l_1

The base case is established by:

$$\begin{array}{lll} & \texttt{map}\, f\, ([] \,@\, l_2) & \\ = & \texttt{map}\, f\, l_2 & def.\ @ \\ = & [] \,@\, \texttt{map}\, f\, l_2 & def.\ @ \\ = & (\texttt{map}\, f\, []) \,@\, (\texttt{map}\, f\, l_2) & def.\ map \end{array}$$

Inductive step: We must establish:

$$\forall xs.\forall x.(P(xs) \Rightarrow P(x :: xs))$$

where the induction hypothesis $P(xs)$ is

$$\texttt{map}\, f\, (xs \,@\, l_2) \;=\; (\texttt{map}\, f\, xs) \,@\, (\texttt{map}\, f\, l_2)$$

Consider arbitrary list xs and element x (of right types). Assume that the induction hypothesis holds. Then, the inductive step is completed by:

$$\begin{array}{lll} & \texttt{map}\, f\, ((x :: xs) \,@\, l_2) & \\ = & \texttt{map}\, f\, (x :: (xs \,@\, l_2)) & def.\ @ \\ = & f(x) :: \texttt{map}\, f\, (xs \,@\, l_2) & def.\ \texttt{map} \\ = & f(x) :: ((\texttt{map}\, f\, xs) \,@\, (\texttt{map}\, f\, l_2)) & ind.\ hyp \\ = & (f(x) :: (\texttt{map}\, f\, xs)) \,@\, (\texttt{map}\, f\, l_2) & def.\ @ \\ = & \texttt{map}\, f\, (x :: xs) \,@\, (\texttt{map}\, f\, l_2) & def.\ \texttt{map} \end{array}$$

By the structural induction principle for lists, we conclude that (1) holds for all lists l_1 and l_2.

Problem B

Consider the declarations:

```
datatype 'a tree = Lf | Br of 'a * 'a tree * 'a tree

fun inorder Lf              = []
  | inorder(Br(x, t1, t2)) = inorder t1 @ (x :: inorder t2)

fun io(LF, xs)             = xs
  | io(Br(x, t1, t2), xs) = io(t1, x :: io(t2, xs))
```

Prove that

$$\texttt{inorder}(t) \,@\, l \;= \texttt{io}(t, l) \qquad\qquad (2)$$

for all $t \in {}'\texttt{a}\,\texttt{tree}$ and $l \in {}'\texttt{a}\,\texttt{list}$. You may assume that `@` is associative.

We prove by structural induction on trees that for all $t \in {}'\texttt{a}\,\texttt{tree}$:

$$\forall l.(\texttt{inorder}(t) \,@\, l \;= \texttt{io}(t, l)) \qquad\qquad (*)$$

The base case is established as follows: Consider an arbitrary list l. Then we have

$$\begin{array}{lll} & \texttt{inorder}(\texttt{Lf}) \,@\, l & \\ = & [] \,@\, l & \textit{def. inorder} \\ = & l & \textit{def. @} \\ = & \texttt{io}(\texttt{Lf}, l) & \textit{def. io} \end{array}$$

For the inductive step, we must establish:

$$\forall t_1, t_2, n.(P(t_1) \wedge P(t_2) \;\Rightarrow\; P(\texttt{Br}(n, t_1, t_2)))$$

where $P(t)$ is

$$\forall l'.(\texttt{inorder}(t) \,@\, l' \;= \texttt{io}(t, l'))$$

Consider arbitrary trees t_1, t_2, and element n (of suitable types). Assume the induction hypotheses $P(t_1)$ and $P(t_2)$. Consider arbitrary list l. The inductive step is completed by:

$$\begin{array}{lll} & \texttt{inorder}(\texttt{Br}(n, t_1, t_2)) \,@\, l & \\ = & (\texttt{inorder}(t_1) \,@\, (n :: \texttt{inorder}(t_2))) \,@\, l & \text{def. inorder} \\ = & \texttt{inorder}(t_1) \,@\, ((n :: \texttt{inorder}(t_2)) \,@\, l) & \text{append is associative} \\ = & \texttt{io}(t_1, (n :: \texttt{inorder}(t_2)) \,@\, l) & \text{ind. hyp. } l' \mapsto ((n :: \texttt{inorder}(t_2)) \,@\, l) \\ = & \texttt{io}(t_1, n :: (\texttt{inorder}(t_2) \,@\, l)) & \text{def. @} \\ = & \texttt{io}(t_1, n :: \texttt{io}(t_2, l)) & \text{ind. hyp. } l' \mapsto l \\ = & \texttt{io}(\texttt{Br}(n, t_1, t_2), l)) & \text{def. io} \end{array}$$

By the structural induction principle for trees, we conclude that (*) holds for all trees t.

Notice that the explicit quantification ($\forall l$) is needed in ($*$) to make the induction hypotheses strong enough. Without this quantification the above inductive step could not be completed. (Make sure that you understand why.)